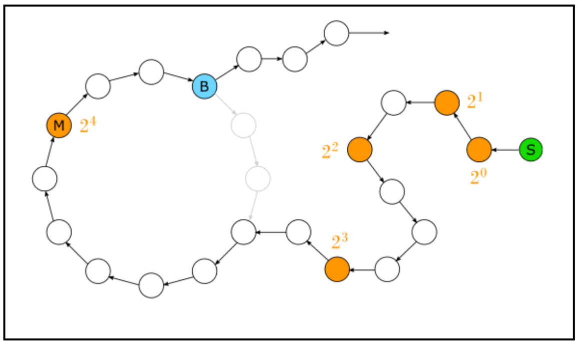

Figure 3. Breaking out of a cycle

breaking out of a cycle should be as small as possible. Also, the decision when to use a different iteration should be simple.

Note that in general, the possible gain in quality is large. If both transition functions were possible and equally likely in each state, then each node in the graph would have two outgoing edges. For randomly chosen edges, our experiments indicate that the graph then is weakly connected, with a strongly connected component that comprises about 84% of the nodes, and a rich inner structure. Let k be the average distance between two break-out states. In this case, k should be chosen small enough that at least one break-out state is on each of the larger cycles.

4.1 Analysis for Logistic Map

Figure 4 shows the analysis results for a break-out to a random node in the state graph for the Logistic Map transition function. As a special case of breaking-out every k steps on average, we are using a counter and break out exactly after k steps, starting with $k = 1$. Experiments have been performed with different values of k, ranging from 2 to 1024. The x axis shows the values of k, while on the y axis the maximum cycle length that was found for a run with 100 different start values is displayed. The random transition function that was used for the selection of the target node was the AES symmetric stream cipher, used in Cypher Feedback (CFB) mode, meaning that the output of the encryption algorithm was used as input for the next iteration. With AES being a highly regarded cryptographic algorithm, this creates a pseudo random generator of exceptional quality, but even more importantly without any expected statistic dependence on the Logistic Map function. The result shows a very mixed behavior: for some values of k, the maximum cycle length is increased compared to the values in Table 2. But e.g. for k=64, the maximum cycle length is even lower than without the break-out mechanism. The conclusion that can be drawn is, that a blind break-out to a random target node will not generally improve the state space structure, but might randomly improve or degrade it, dependent on the chosen target node.

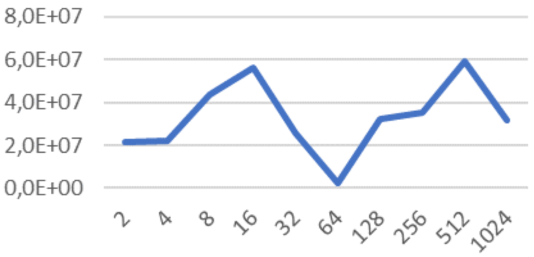

Figure 4. Maximum cycle length for Logistic Map using random break-out target nodes

5. Parameter Modification

A purely random approach to the break-out method apparently does not necessarily have a positive impact, as was shown in the previous section. Furthermore, implementing a statistically independent pseudo random generator in addition to the original transition function does not appear very efficient from a computational point of view, leading to a significantly increased required chip area for hardware implementations. So instead of approximating a random function with expected values as given by [7], we propose to use the chaotic function with a different parametrization instead. The properties of chaotic functions should ensure that the changed parametrization leads to a behavior that is statistically independent enough from the original function.

5.1 Analysis for Logistic Map

Looking at the Logistic Map function $f_1(x) = a \cdot x \cdot (1 - x)$, there is not much potential for parametrization: the only candidate to modify is parameter a.

The analyzed approach was to switch between a value of 3.99 and 3.98 every k steps with k ranging from 2 to 1024. Table 3 shows the result of the state space analysis. It can be seen that breaking out of the cycles increases the maximum cycle length significantly by a minimum factor of 2 for $k = 2$ up to a factor of 227 for $k = 1024$. It can also clearly be seen that the maximum cycle length increases with k. Figure 5 shows this relationship. The graph is not completely consistent with k-values of 32 and 256 showing a decline compared to the previous values.

This can most probably be explained by the very low number of 10 start values for measuring the cycle lengths.

Table 3. Analysis results of Logistic Map Switching to alternative A-parameter every k steps

k	Cycle Lengths	Maximum Tree Heights	% of Start Values
2	14258373	169347594	100
4	81839355	219447222	100
8	92330073 158592654 23977917 5583933	116649348 153814118 39516082 29542408	50 20 20 10
16	166687958 18456033	347193629 66490210	90 10
32	16463304 106561026	509498228 184708805	50 50
64	285351755 27448135 186333745	247720291 163920805 138538519	70 20 10
128	365089092 490073193	459489459 568127512	90 10
256	122507017 61896137	1011052679 766210246	50 50
512	918541890 505315773	913649910 506390293	70 30
1024	1517222425 997359850	2056217674 1168909136	60 40

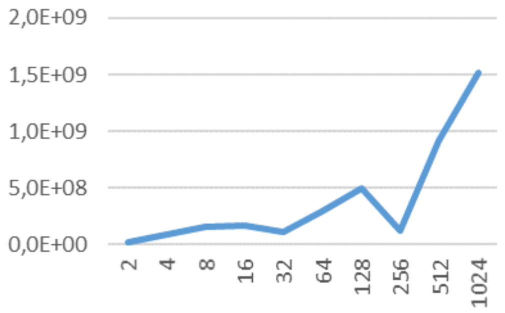

Figure 5. Maximum cycle length over *k* for Logistic Map

5.2 Analysis for Trigonometric Function

Similar investigations have been performed for the Trigonometric function. Again, the function $f_2 = \sin^2(z \cdot \arcsin\sqrt{x})$ has not many options to be parametrized. The most obvious one is a modification of the z parameter. The analyzed modification is a switch between the two z-values 2 and 3 after k iterations with k ranging from 2 to 1024. Table 4 shows the analysis results. Again, the maximum cycle length was increased for all k-values except for $k = 2$ by factors between 7.4 and 75. Figure 5 shows the relationship between k and the maximum cycle length and shows a similar dependency as for the Logistic Map function.

Table 4. Analysis results of Trigonometric Function switching to alternative z-parameter every k Steps

k	Cycle Lengths	Maximum Tree Heights	% of Start Values
2	7759233 7483845 6666741	79946065 8203387 51758320	90 10 10
4	72851705 25009350	101640527 36265720	90 10
8	149450121 18636894	137849328 13201151	90 10
16	132378065	135672835	100
32	380555241	491267040	100
64	60296210 120161535 63678680 191870250	35947884 89028321 25751263 288875294	10 10 10 70
128	182362011 8997879	418726507 244129755	80 20
256	561225035 305935113	231801614 373691537	50 50
512	633619125 218623158 218623158	672871645 662621246 508761773	40 20 40
1024	728877500 251601625 155949650	986558422 529695505 66719039	30 30 40

6. Statistical evaluation

While the positive impact of the break-out method could be demonstrated in the previous section, it seems worthwhile to verify that the change to the transition function does not have a negative impact on its statistical behavior. One of the most commonly used methods to evaluate the statistical properties of pseudo random number generators is the NIST test battery. It comprises 15 different statistical tests that each result in a number of passed and failed test cases. For a “good” PRNG, the proportion of passed to failed test cases is expected to be greater than 96% for each of the tests. Figure 7 shows the number of passed test cases per test, sorted in increasing order for easier comparability. The blue dotted line is the result for the original unmodified Logistic Map and clearly indicates, that using this algorithm as cryptographic pseudo random number generator without any further modification cannot be recommended due to the significant number of tests with less than 96 passing test cases. The red solid line is the result for the Logistic Map using a regular break-out with *k*=1024. The same modified parameterization as presented in Section 5.2 has been used. It can easily be seen that the result of the NIST test suite is comparable to the result for the unmodified algorithm, so the break-out method does not have a negative impact on the statistic behavior of the algorithm in this case.

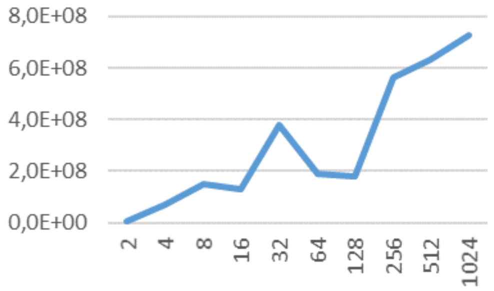

Figure 6. Maximum cycle length over *k* for Trigonometric function

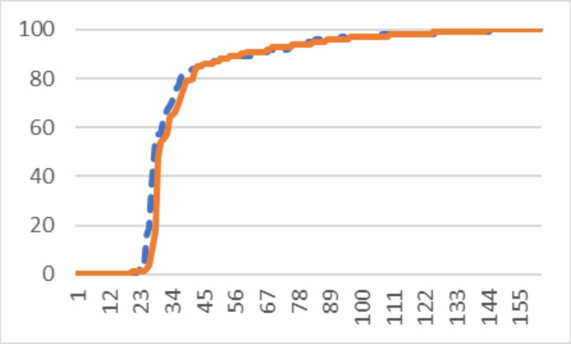

Figure 7. Maximum cycle length over *k* for Trigonometric function

The same analysis has been performed on the original and modified Trigonometric function, using *k*=1024 and the same parameterization as in Section 5.2. The result shown in Figure 8.depicts, that in this case the result of the NIST test battery is even a bit better for the break-out version of the algorithm than for the original function.

From the analysis above it can be concluded that the break-out method does not affect the statistic properties of the examined chaotic functions in a negative way.

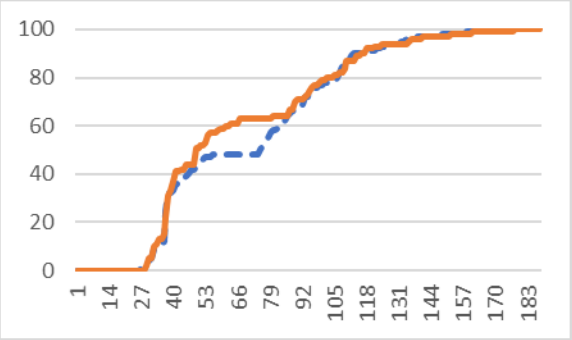

Figure 8. Maximum cycle length over *k* for Trigonometric function

7. Conclusion

A simple method to avoid short cycle lengths in implementations of PRNGs based on chaotic functions was presented. It could be demonstrated that the cycle length for the Logistic Map function can be extended significantly by modifying the parameterization of the chaotic function for certain iterations. This might make chaotic PRNGs usable for an extended range of security applications where increasing the size of the state is not an option because of hardware or computational restrictions. One might even think about this method as a possibility to “repair” an already built-in weak PRNG (even in hardware), as the second transition might be realized in the form of a re-seeding. This approach leads to significant improvements for both investigated chaotic functions (Logistic Map and Trigonometric function).

Further improvements can potentially be achieved by hardcoding an extra transition. In the case of several components, the method of extra transitions could even be extended to link all components together, so that for *all* seed values, a larger cycle length is guaranteed.

Due to the low computational complexity, chaotic algorithms should be investigated further, e.g. in the context of RFID with its limitations on chip area and energy consumption.

As future work, it seems important to gather more statistically relevant data by analyzing a higher number of start values, e.g. on a high-performance computer. Furthermore, extending the investigations towards fix point implementations of chaotic functions, that according to our preliminary experiments seem to exhibit a better behavior than floating point implementations, seems a reasonable way forward.

Finally, the structure of graphs where each node has two outgoing edges might be investigated.

[1] M. Abutaha et al., “Design of a peudo-chaotic number generator as a random number generator”, in International Conference on Communications (COMM), 2016.

[2] A. Beckmann, J. Fedorowicz, J. Keller, and U. Meyer, “A structural analysis of the a5/1 state transition graph,” in First Workshop on GRAPH Inspection and Traversal Engineering, ser. Electronic Proceedings in Theoretical Computer Science, vol. 99. Open Publishing Association, 2012, pp. 5–19.

[3] A. Biryujkov, A. Shamir, and D. Wagner, “Real Time Cryptanalysis of A5/1 on a PC”, in Proceedings of Fast Software Encryption 7th International Workshop, New York, 2000.

[4] A. Desai, A. Hevia, and Y. L. Yin. "A practice-oriented treatment of pseudorandom number generators." International Conference on the Theory and Applications of Cryptographic Techniques. Springer Berlin Heidelberg, 2002.

[5] Y. Dodis, D. Pointcheval, S. Ruhault, D. Vergniaud, and D. Wichs, “Security analysis of pseudo-random number generators with input:/dev/random is not robust.” , Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, ACM 2013.

[6] K. Entacher, “A collection of selected pseudorandom number generators with linear structures,” ACPC-Austrian Center for Parallel Computation, Tech. Rep., 1997.

[7] P. Flajolet and A. M. Odlyzko, “Random mapping statistics,” in Advances in Cryptology. Springer Verlag, 1990, pp. 329–354.

[8] J. Golic, “Cryptanalysis of Alleged A5 Stream Cypher”, in Proceedings of Advances in Cryptology – Eurocrypt 97, Konstanz, 1997.

[9] M. Hamdi, R. Rhouma, and S. Belghith, "A very efficient pseudo-random number generator based on chaotic maps and S-box tables." Int. J. Comput. Control Quantum Inform. Eng 9 (2015), pp. 481-485.

[10] J. Keller, “Parallel exploration of the structure of random functions,” in Proceedings of the 6th Workshop Parallele Systeme und Algorithmen (PASA) in conjunction with the International Conference on Architecture of Computing Systems, ARCS. VDE, 2002.

[11] J. Keller, H. Wiese, “Period lengths of chaotic pseudo-random number generators.” in Proceedings of the Fourth IASTED International Conference on Communication, Network and Information Security. pp. 7-11. CNIS '07, ACTA Press, Anaheim, CA, USA, 2007, http://dl.acm.org/citation.cfm?id=1659141.1659144, Access Date: 1st October, 2016.

[12] D. E. Knuth, “Mathematical analysis of algorithms.” in Proc. of IFIP Congress 1971, Information Processing 71. pp. 19-27. North-Holland Publ. Co., 1972.

[13] Z. Kotulski, J. Szczepanski, J., K. Górski, A. Górska, A. Paszkiewicz, “On constructive approach to chaotic pseudorandom number generators.” in Proc. Of RCMCIS 2000, Zegrze. pp. 191-203, 2000.

[14] C. Manifavas, G. Hatzivasilis, K. Fysarakis, K. Rantos, “Lightweight cryptography for embedded systems - a comparative analysis.” in Proc. 8th International Workshop on Data Privacy Management and Autonomous Spontaneous Security, pp. 333-349, Springer, 2014, http://dx.doi.org/10.1007/978-3-642-54568-9_21, Access Date: 1st October, 2016.

G. Marsaglia, “The marsaglia random number cdrom including the diehard battery of tests of randomness,” 1995. [Online]. Available:http://www.stat.fsu.edu/pub/diehard/, Access Date: 1st October, 2016.

[15] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, Handbook of Applied Cryptography. CRC Press, 1996.

[16] B. Mennink, B. Preneel, “On the XOR of multiple random permutations.” in Proc. Applied Cryptography and Network Security. pp. 619-634, 2015.

[17] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, “Statistical test suite for random and pseudorandom number generators for cryptographic applications: Special publication 800-22, revision 1a,” 2010. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800022rev1a.pdf, Access Date: 1st October, 2016.

[18] R. Sedgewick, P. Flajolet, “An Introduction to the Analysis of Algorithms.”, Addison-Wesley, Reading Mass., 1996.

[19] J. Szczepanski, Z. Kotulski, “Pseudorandom number generators based on chaotic dynamical systems.”, Open Systems & Information Dynamics 8(2), 137-146 (Jun 2001), http://dx.doi.org/10.1023/A:1011950531970, Access Date: 1st October, 2016.